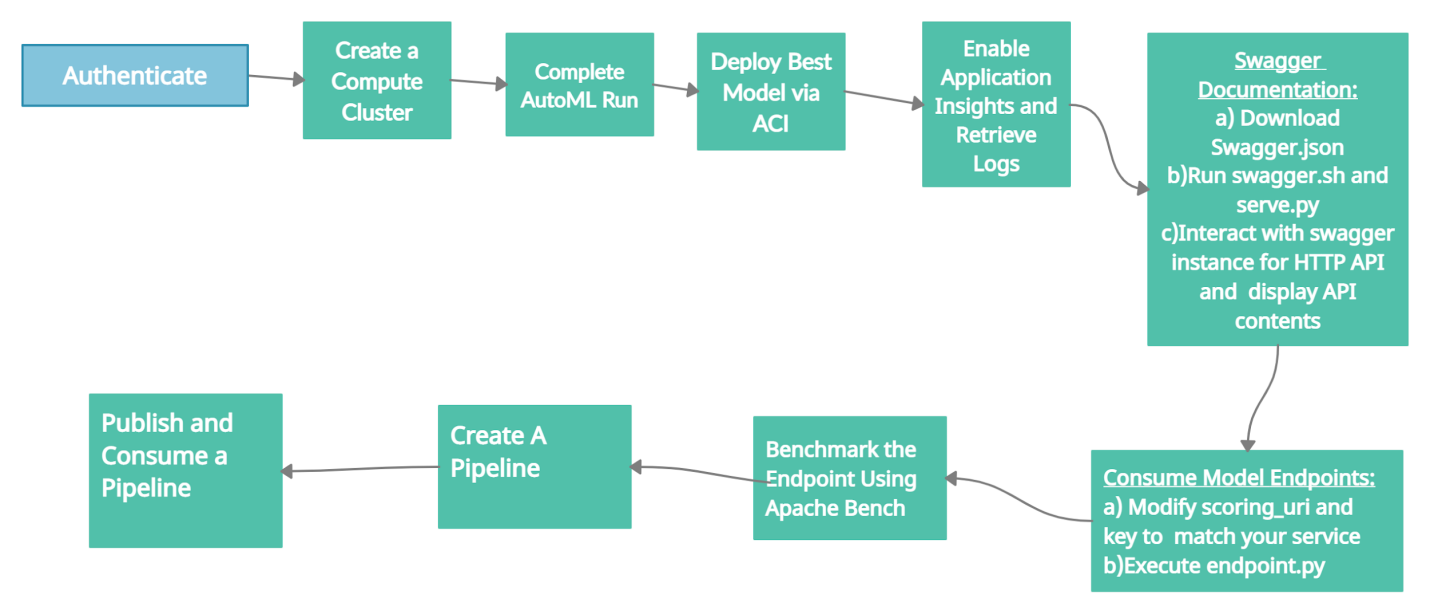
# **Operationalizing Machine Learning**

**Overview:** This project uses a the [Bank Marketing dataset](https://automlsamplenotebookdata.blob.core.windows.net/automl-sample-notebook-data/bankmarketing_train.csv), which contains marketing campaign on phone calls, with input features-both categorical and numerical- like age, job, marital, education, housing,loan, contact, month and campaign etc. The output label 'y' represents whether the marketing campaign was successful or not.

We will use Microsoft Azure to set-up a cloud-based machine learning production model, deploy and consume it. We will also create, publish, and consume a pipeline.

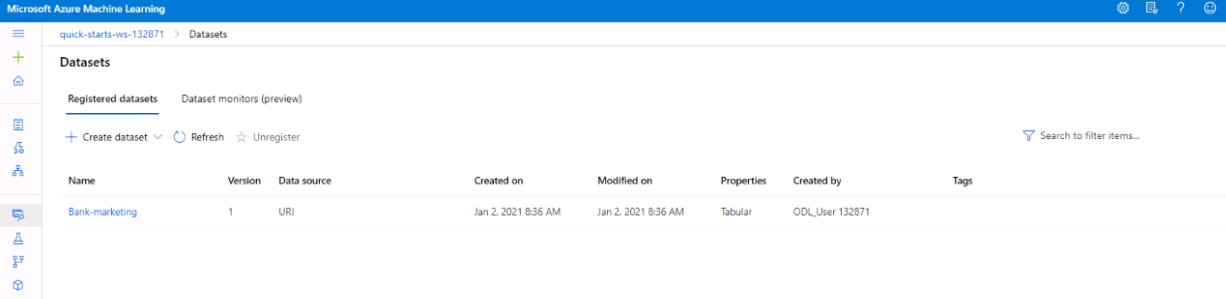
**Architectural Diagram:**



**Key Steps:**

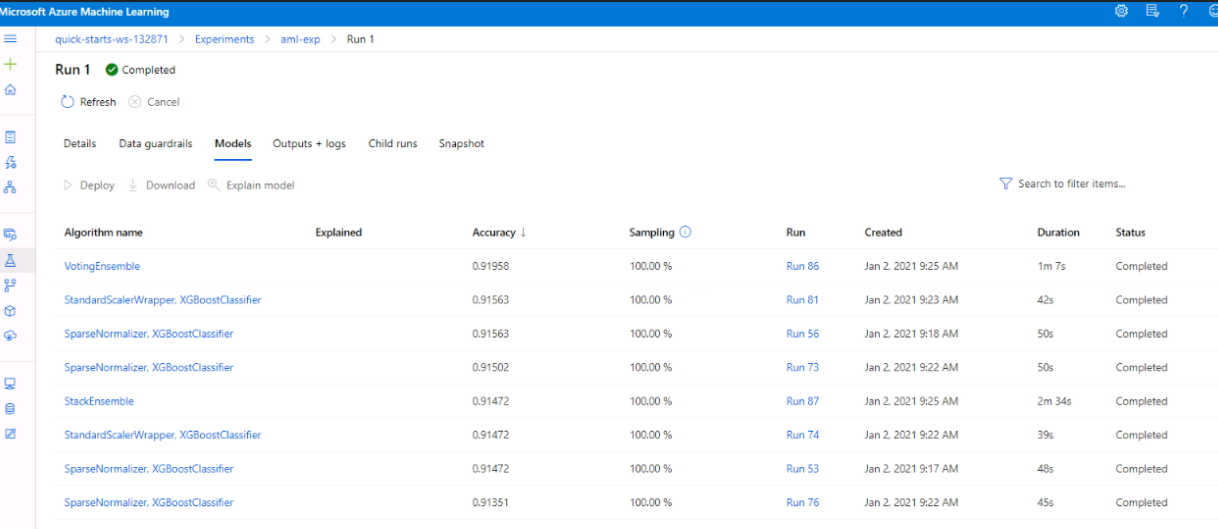
**The main steps are explained below:**

1. **Authentication:** Authentication is crucial for the continuous flow of operations. Continuous Integration and Delivery system (CI/CD) rely on uninterrupted flows. When authentication is not set properly, it requires human interaction and thus, the flow is interrupted. This step is skipped here.
2. **Data Preparation:** The dataset is uploaded and available in the **“registered”** datasets section in the Azure ML Studio.

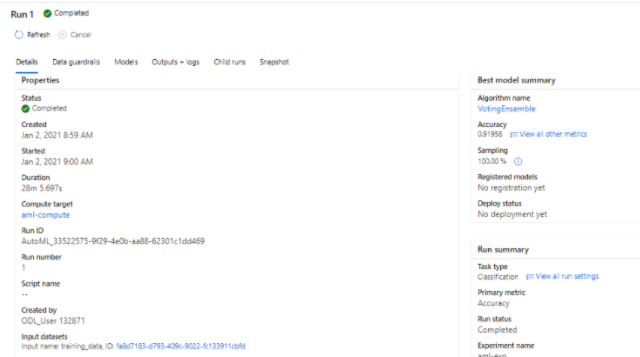


1. **Automated ML Experiment:** An AutoML experiment is run to find the best possible solution for our problem, and choose the top performing model.

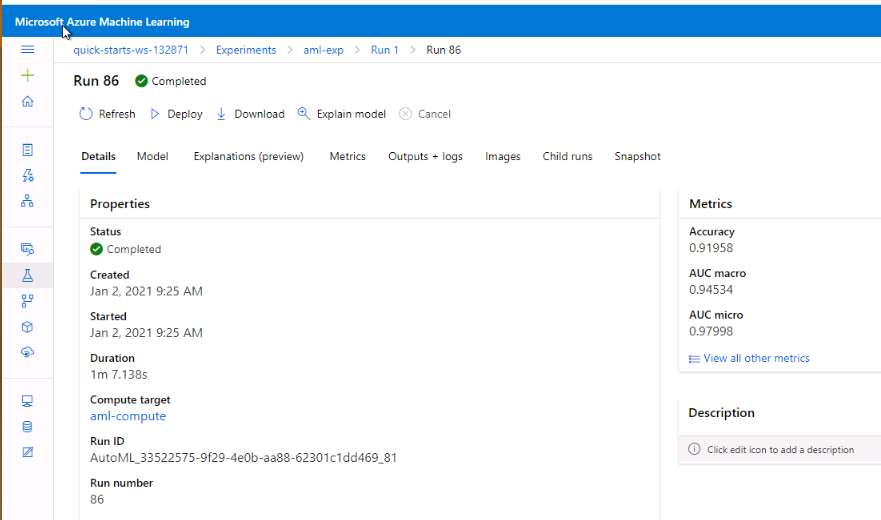
**Results of AutoML Run:** AutoML Run takes about half an hour and provides us with a list of all the top performing models as shown below.

****

The screenshot below shows the completed AutoML run and the best model summary.



Further details of the best model **“Voting Ensemble”**)are shown below.

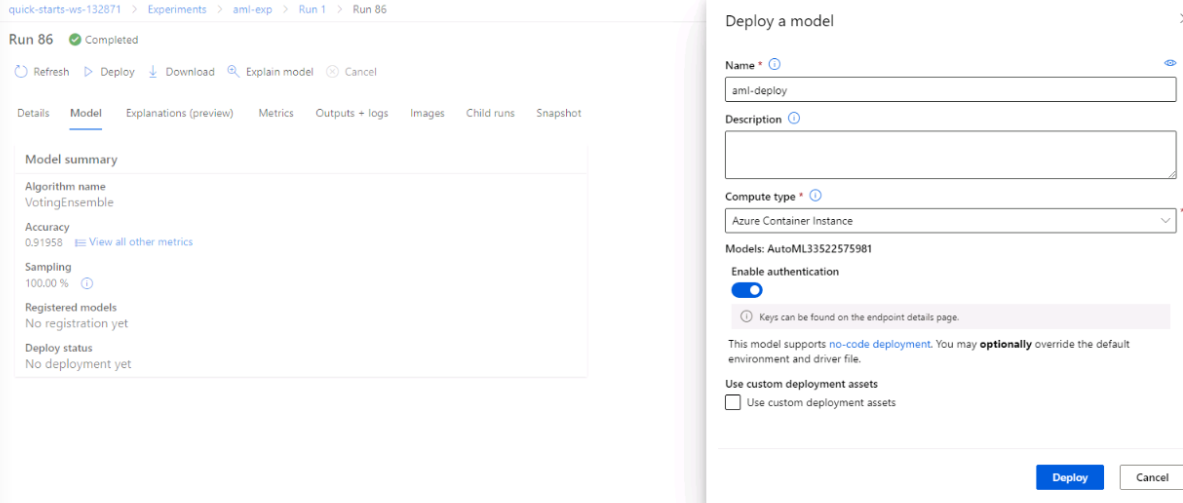


1. **Deploy the best model:**The primary task as a Machine Learning engineer is to ship models into production. Constant evaluation allows identifying potential issues and creating a baseline so that adapting or updating is possible.

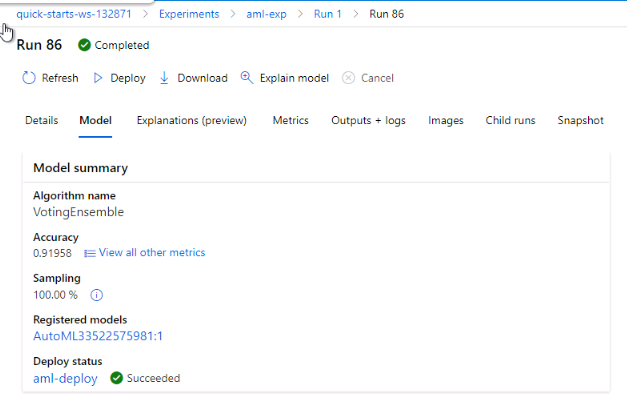
Some key steps to deploy a model are:

* A previously trained model
* Complete the deployment form
* Enable authentication
* Select or create a new compute cluster

The best model is deployed via Azure Container Instances with the name “aml-deploy”.



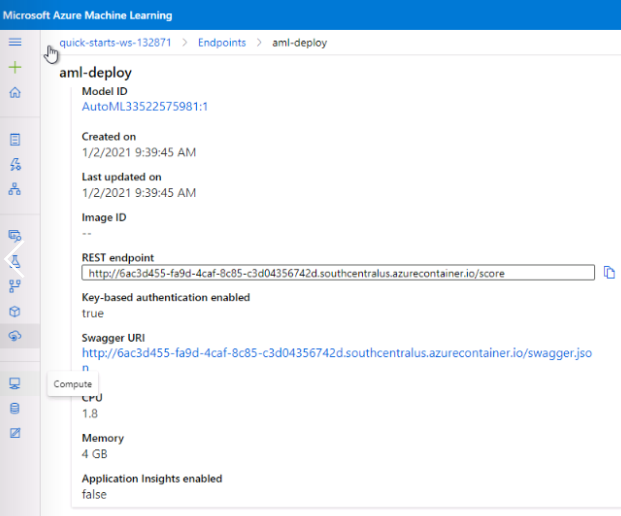
Once the deployment succeeds, the deployment status changes to “Succeeded” as shown below.

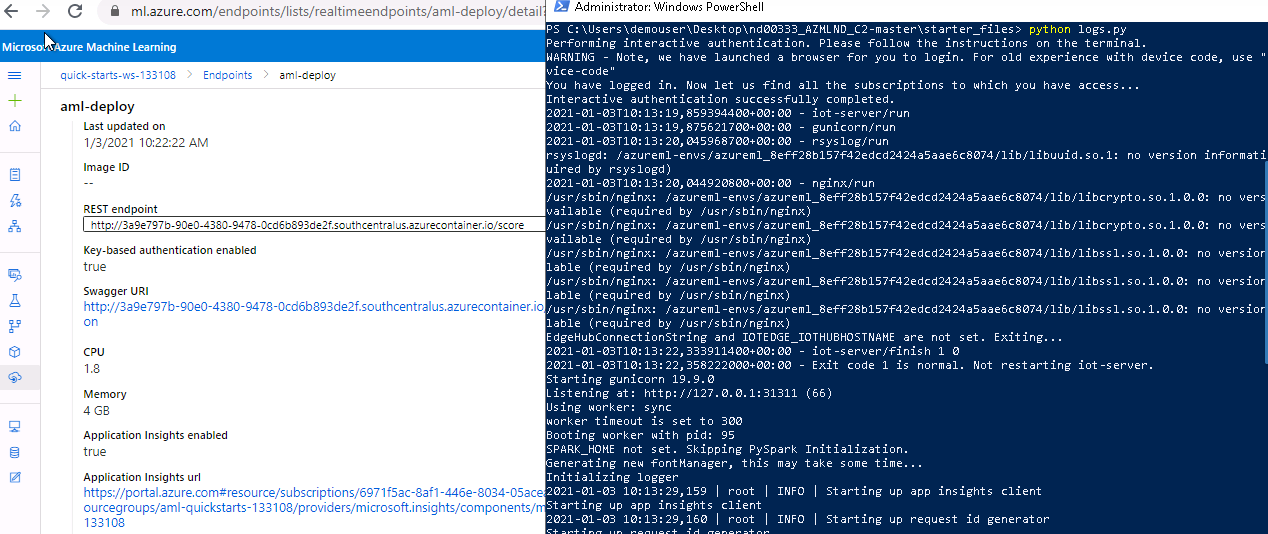
Application Insights that is a very useful tool to detect anomalies, visualize performance. It can be enabled before or after a deployment. T

1. **Enable logging:** Application Insights is a very useful tool to detect anomalies, visualize performance. It can be enabled before or after a deployment**.**

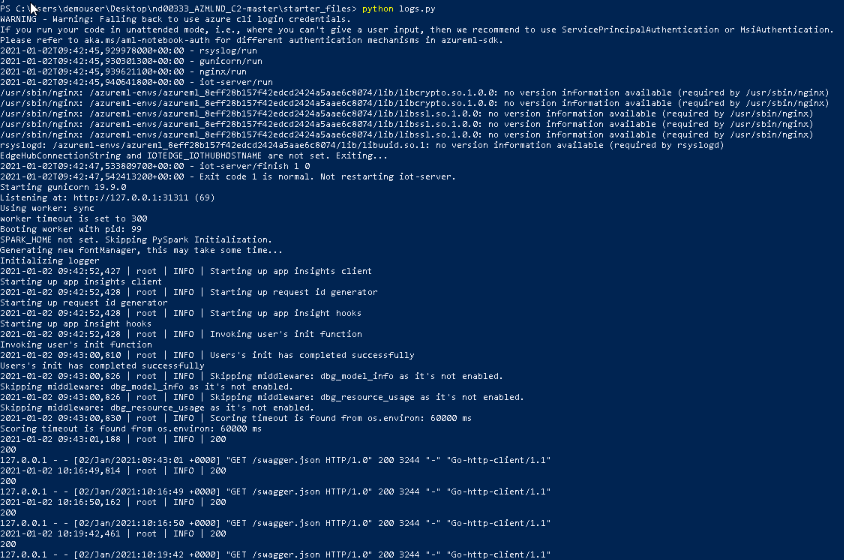
Here, the Application Insights are enabled after deployment and logs(Informational output produced by the software) are retrieved from a deployed model. Enabling Application Insights is vital to debug problems in production environments.

Application Insights are disabled here, so they can be enabled by setting the application insights to true in the logs.py file.

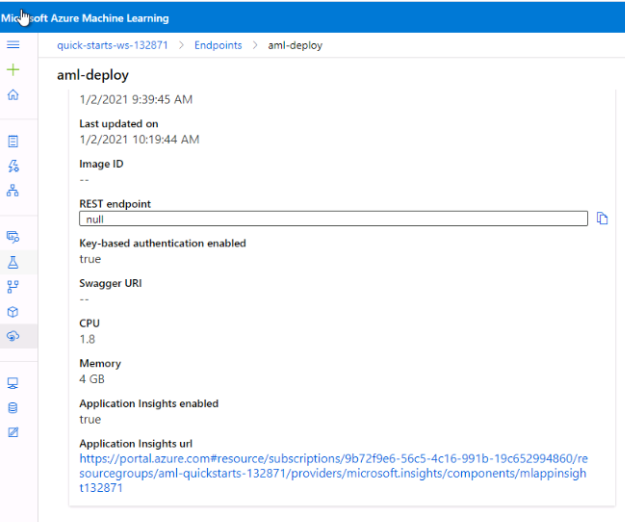


The screenshots below show the output of running “python logs.py” in the terminal.

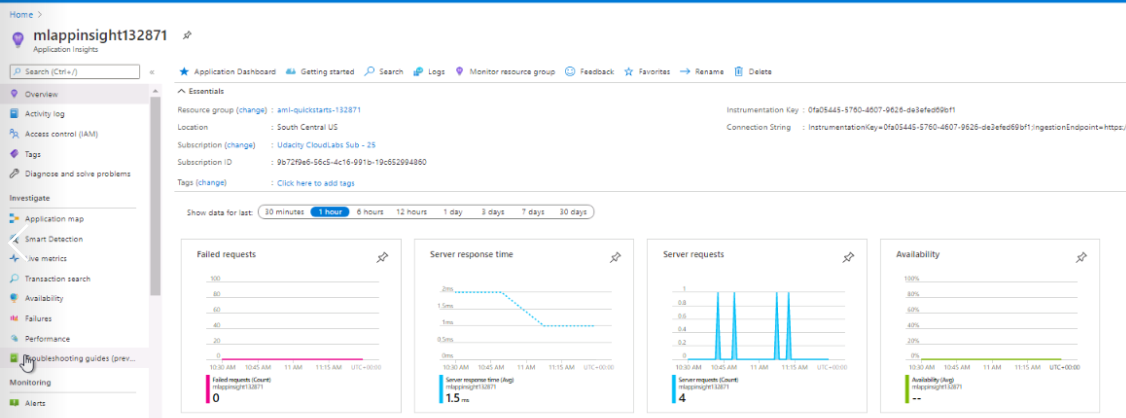
The detailed output of running “python log.py” is shown below, and we can see that application insights client is started.



This enables the application insights as shown below



Following the application insights URL takes us to this screen where the server requests and response time for the past period of time are shown.

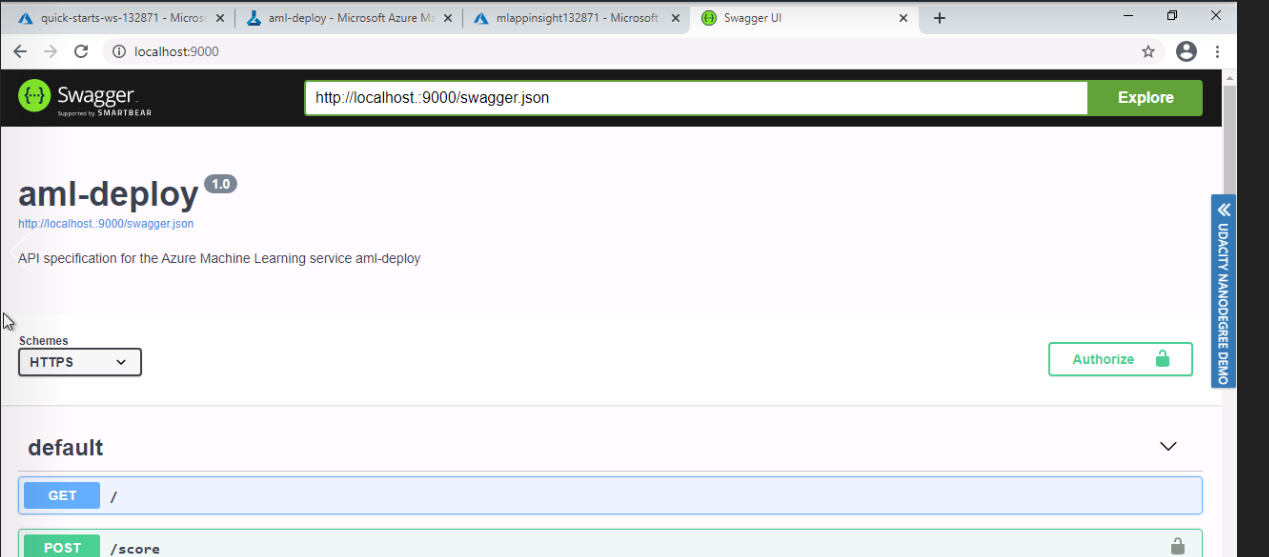


1. **Swagger Documentation and Endpoint Consumption:**

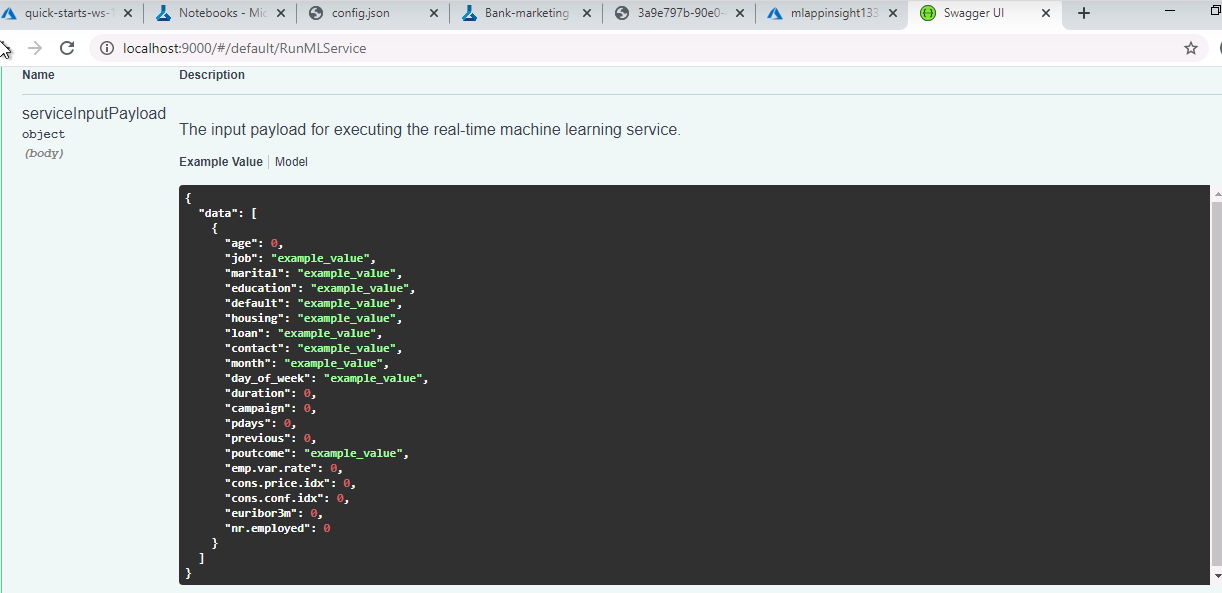
Swagger is a tool that helps build, document, and consume RESTful web services like the ones being deployed in Azure ML Studio . Azure provides a swagger.json file that is used to create a website that documents the HTTP endpoint for a deployed model.

The swagger.json file is downloaded from the Azure ML Studio, and 'swagger.sh' downloads the latest Swagger container, which will run on port 9000. The 'serve.py' will start a Python server on port 8000.

Going to http://local host:9000 and typing the address of swagger.json file gives the result below.

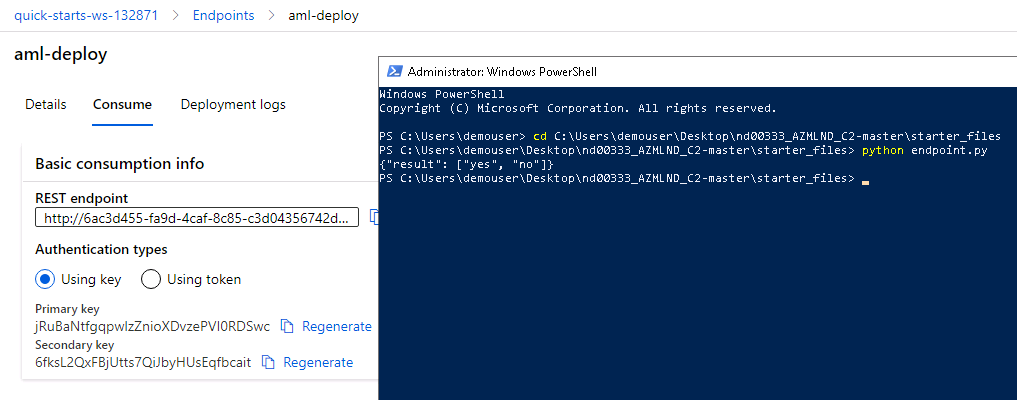


We get a simplified document to consume the HTTP API. Swagger runs on localhost showing the HTTP API methods and responses for the model. This shows the POST HTTP Request section.

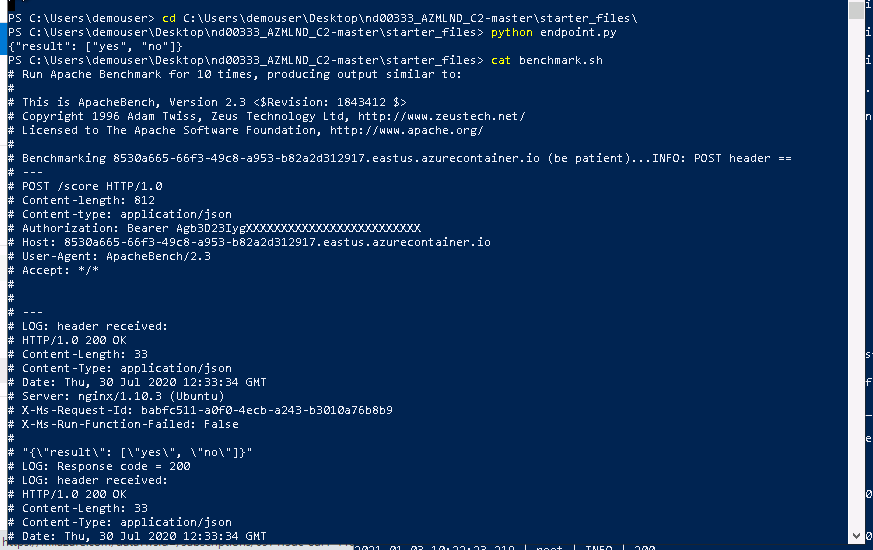


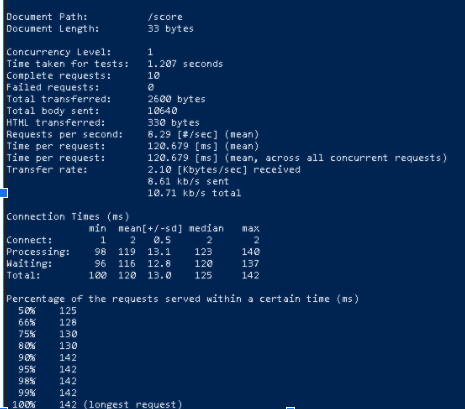
:

After deploying the model, we use the updated 'endpoint.py' script to interact with the trained model and get the following result(either yes or no, which we have predicted.)



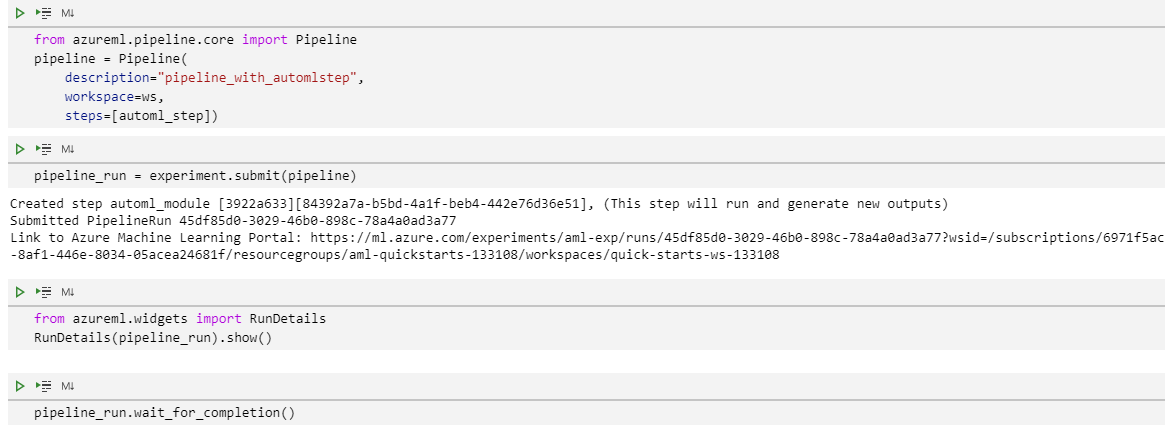
**Benchmarking:** A 'data.json' file is created as response to HTTP REST API endpoint. Apache Benchmarking is used to benchmark the deployed model. We run benchmark.sh file after using updates key and scoring\_uri to match our current service, and get the following result which shows the time taken for test, the completed requests(10), failed requests(0) and time taken per request and percentage of requests served within a certain time.

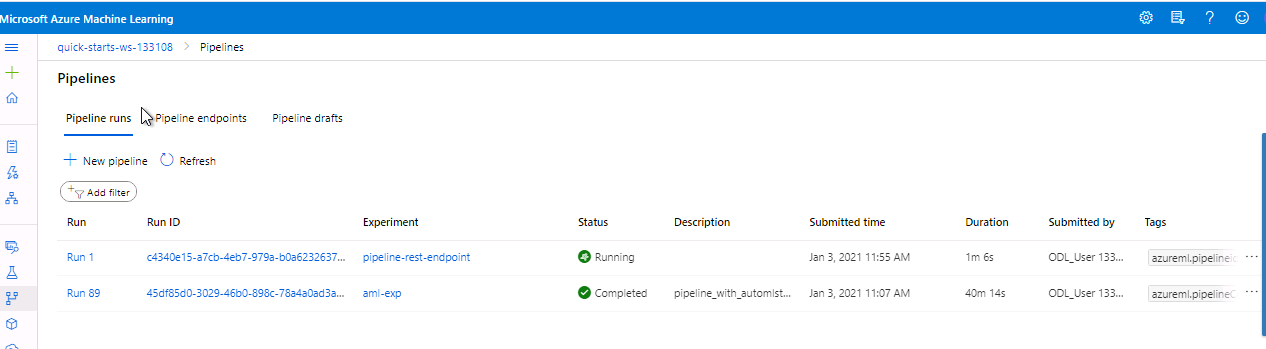




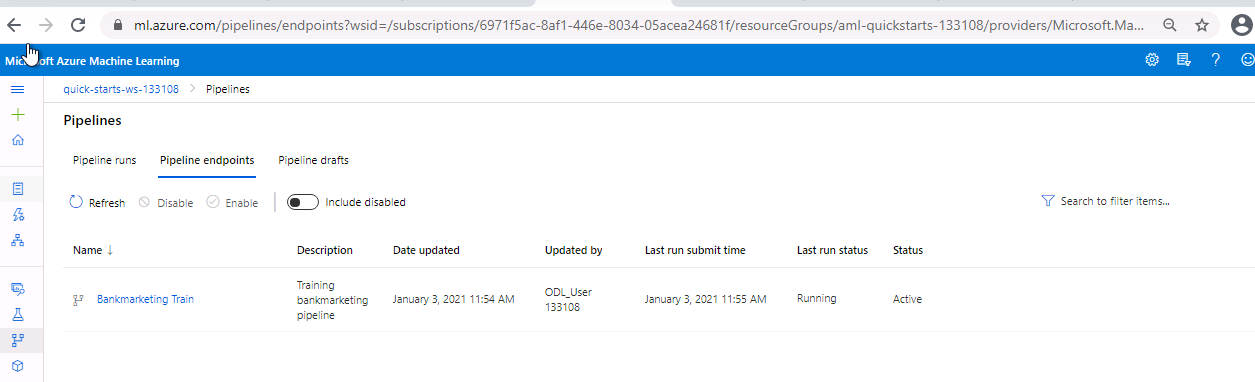
1. **Create and publish a pipeline:** We will use the Jupyter Notebook provided in the starter files and update the notebook to have the same keys, URI, dataset, cluster, and model names already created. Running all the cells will create, publish and consume a pipeline.

**Pipeline Creation:** A great way to automate workflows is via Pipelines. Published pipelines allow external services to interact with them so that they can do work more efficiently**.** Pipelines can take configuration and different steps, like AutoML for example.Pipeline is created here by using the Pipeline class and setting its parameters in Jupyter notebook.

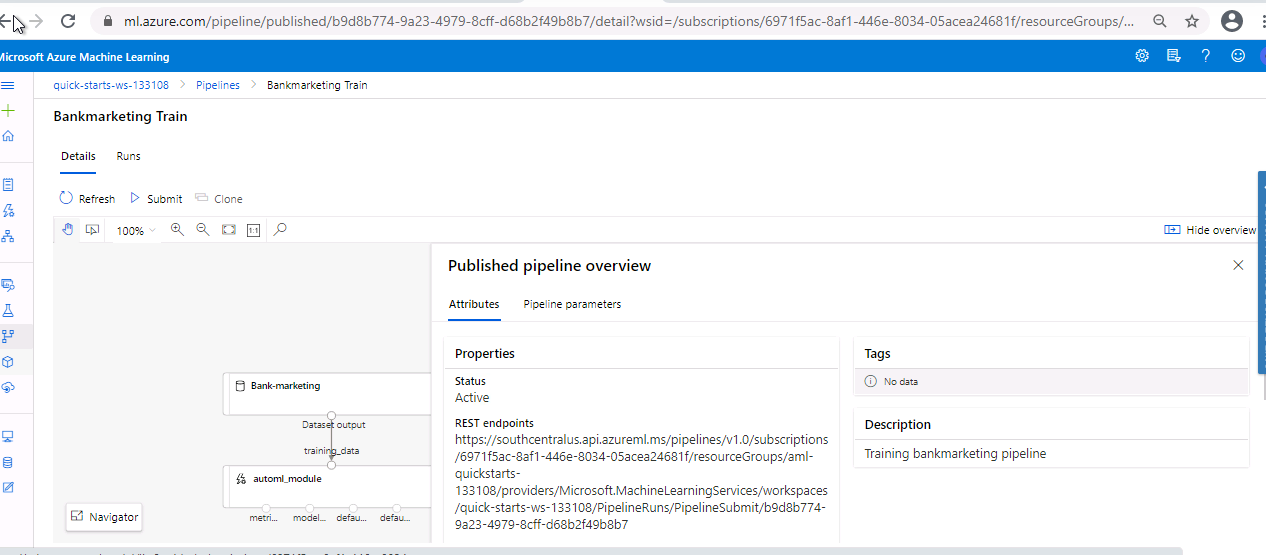


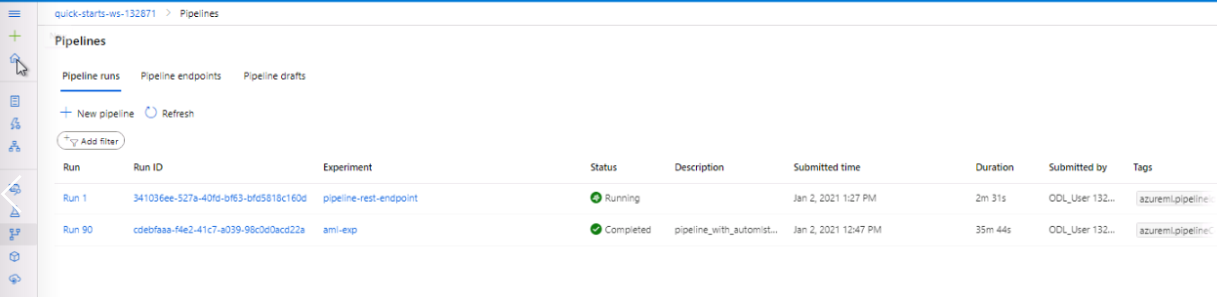
****

**Pipeline Endpoint:** This shows that the endpoint has been created for the pipeline.

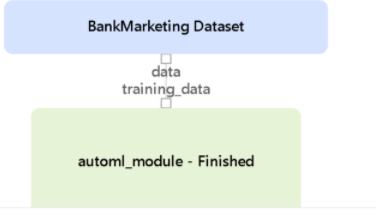
****

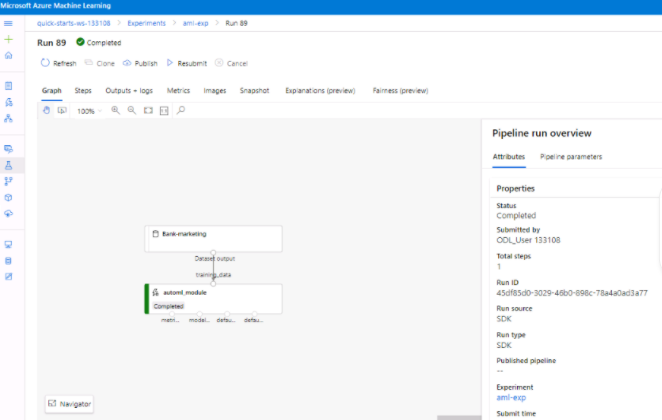
**Publish and Consume the Pipeline:** When a Pipeline is published, a public HTTP endpoint becomes available, allowing other services, including external ones, to interact with an Azure Pipeline.This screenshot shows the attributes of the published pipeline.

****

****

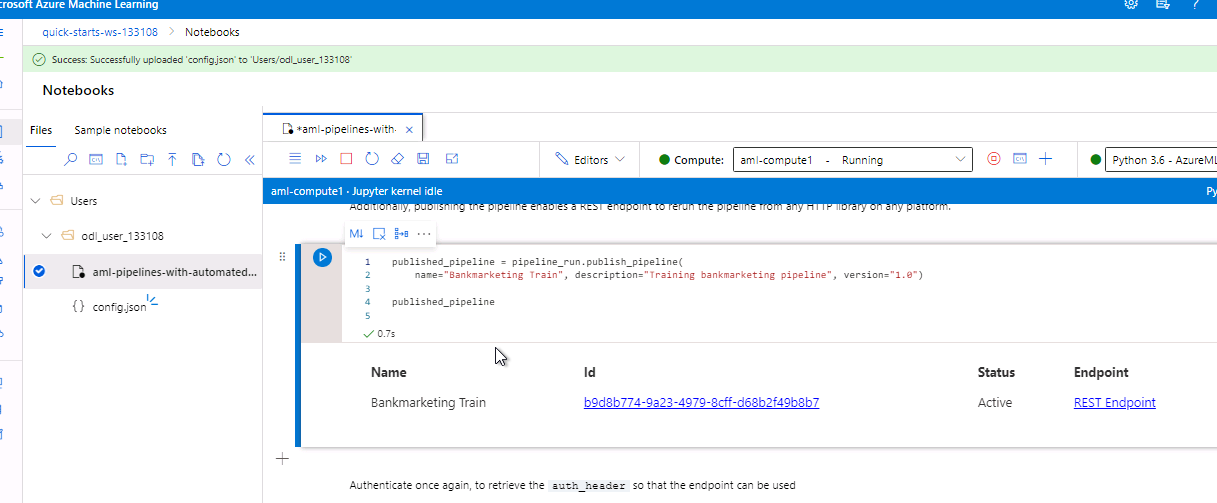
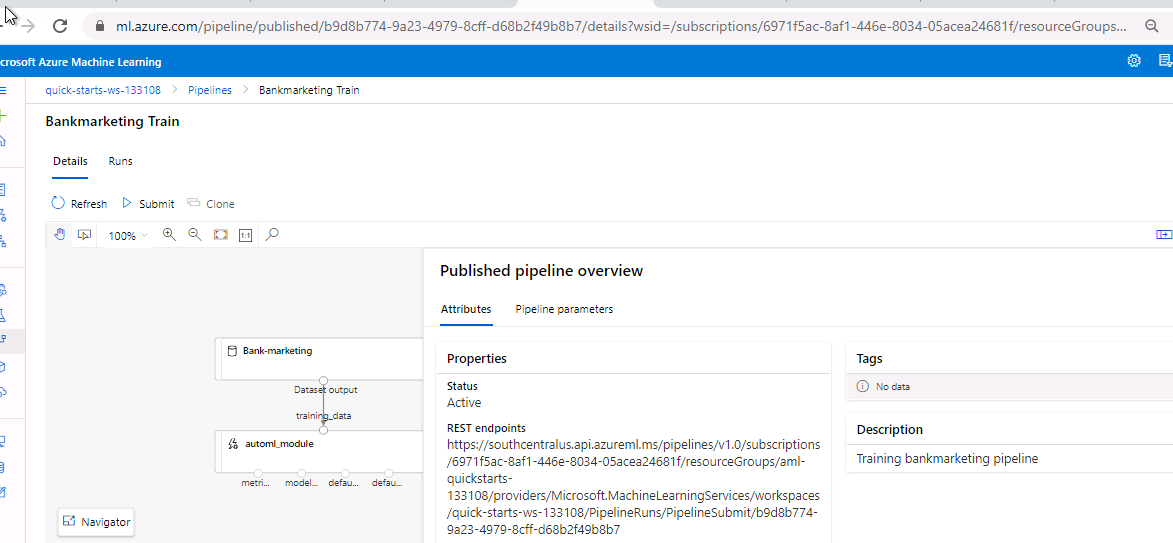
**Bank Marketing dataset with the AutoML module:** The screenshots below show the dataset with AutoML module.

****

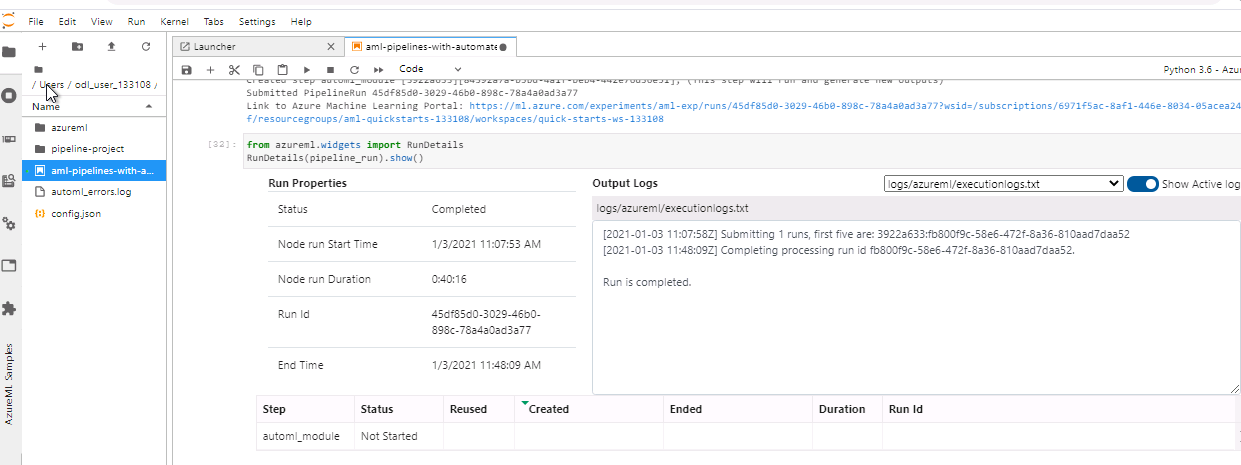
****

**Publish and run from a REST Endpoint:** Publishing the pipeline enables a REST endpoint to rerun the pipeline from any HTTP library on any platform.

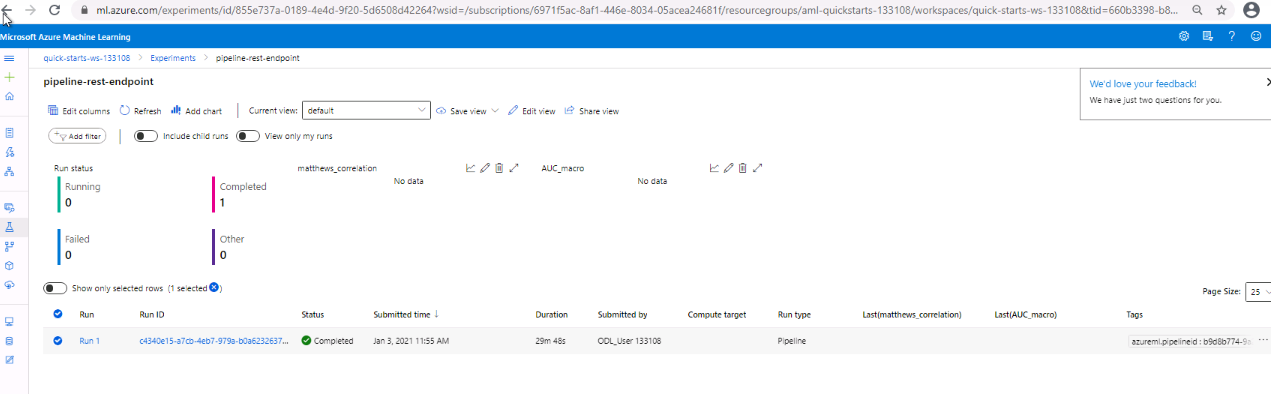
The status of the REST endpoint is shown as “Active’ in screenshots below.

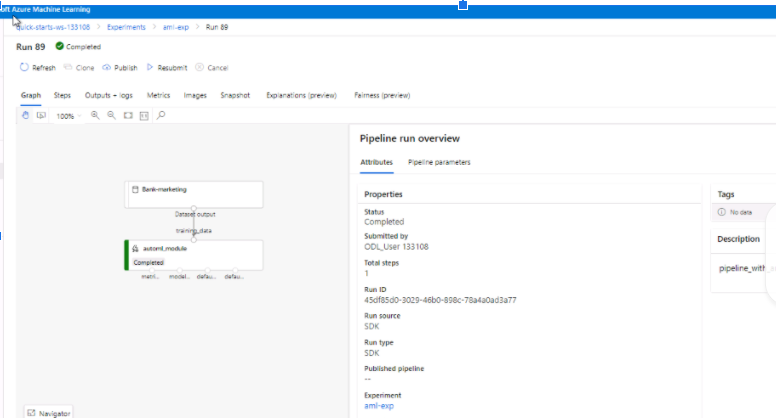
****

**Run Widgets:** The following screenshots show the result of RunDetails function. We can see the start and end times of the experiment, along with the Run ID and output logs.

****

**Completed Run:** Now, the pipeline run has been shown to be completed below.

****

****

**Screen Recording:**

<https://drive.google.com/file/d/1NHM3rQWGS460G7oUVIDlLDFnQIQNd4SP/view>

**Suggestions:**

* AutoML gives a warning due to presence of imbalance dataset in the beginning, this can be rectified by first removing the imbalance by using techniques like SMOTE. More details on this can be found [here](https://towardsdatascience.com/how-to-handle-smote-data-in-imbalanced-classification-problems-cf4b86e8c6a1#:~:text=SMOTE%20(Synthetic%20Minority%20Over%2Dsampling,making%20small%20changes%20to%20them.).
* We can change the timeout to more than 1 hour to get better performing models.